

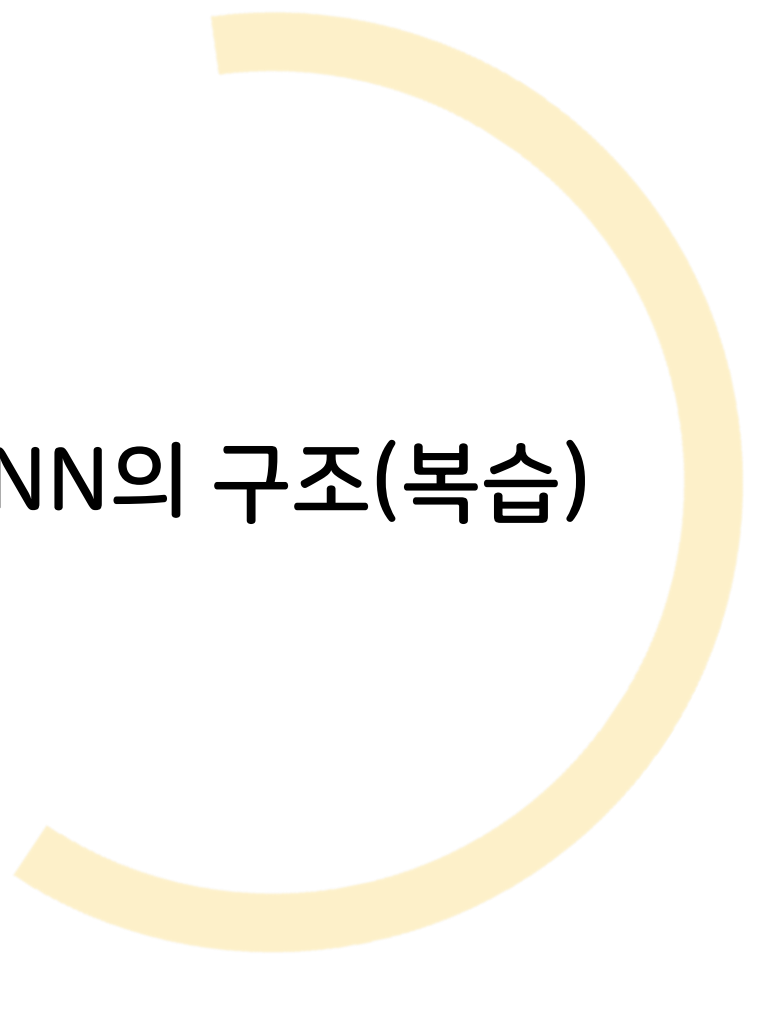
parrot Data Science

8th Session – Convolutional Neural Network

Contents



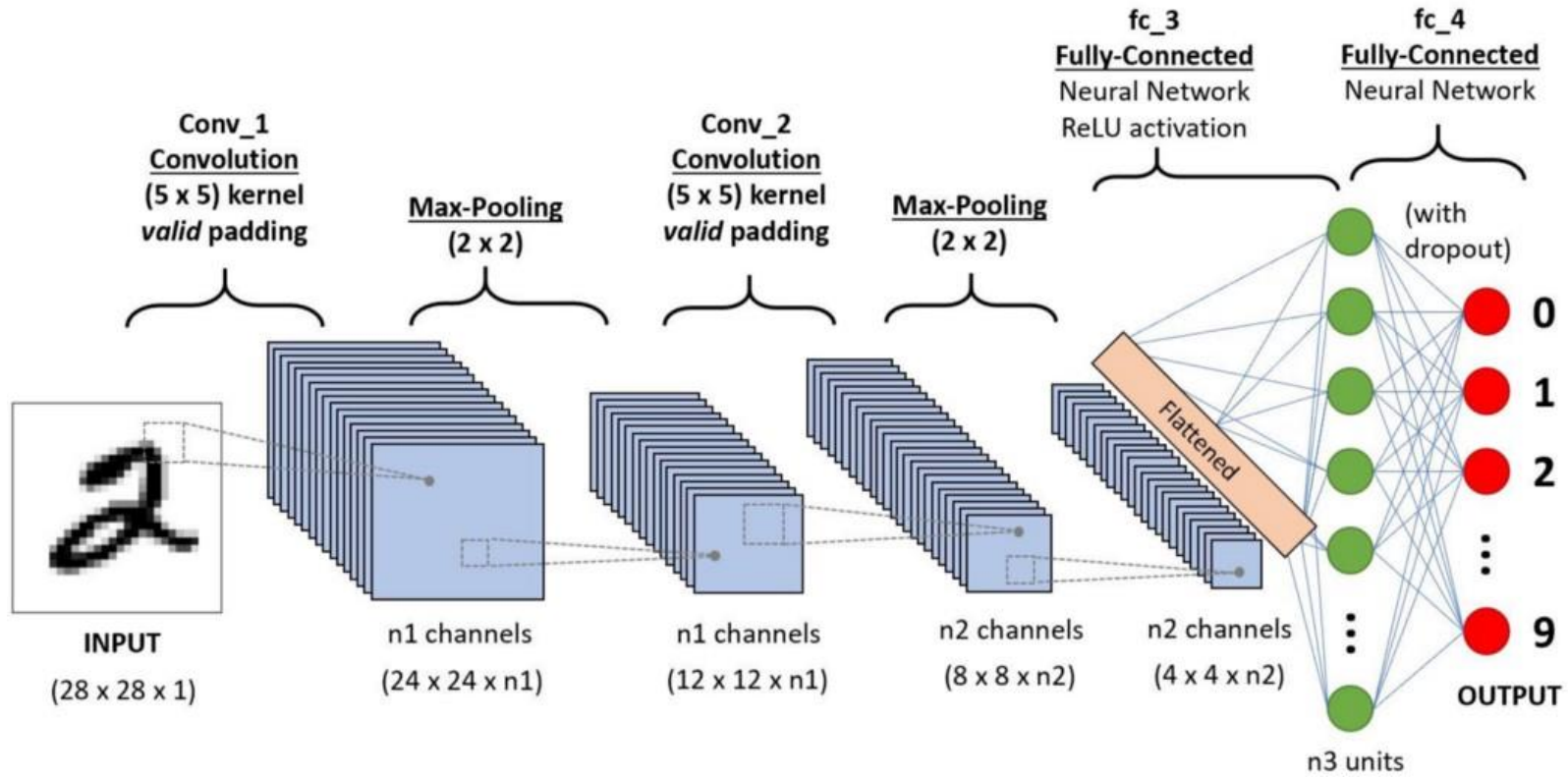
- CNN의 구조(복습)
- Components of CNN
- 모델의 최적화
- 실습



CNN의 구조(복습)

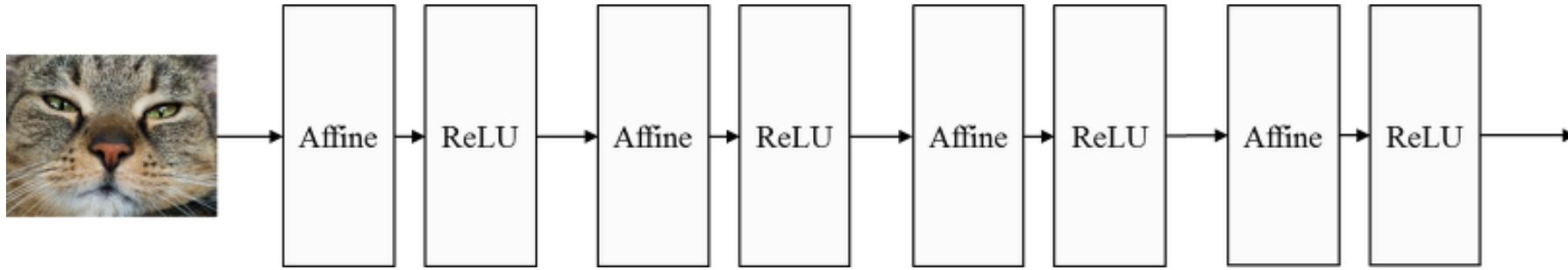
CNN의 구조적 특징 overview

CNN의 구조 (복습)



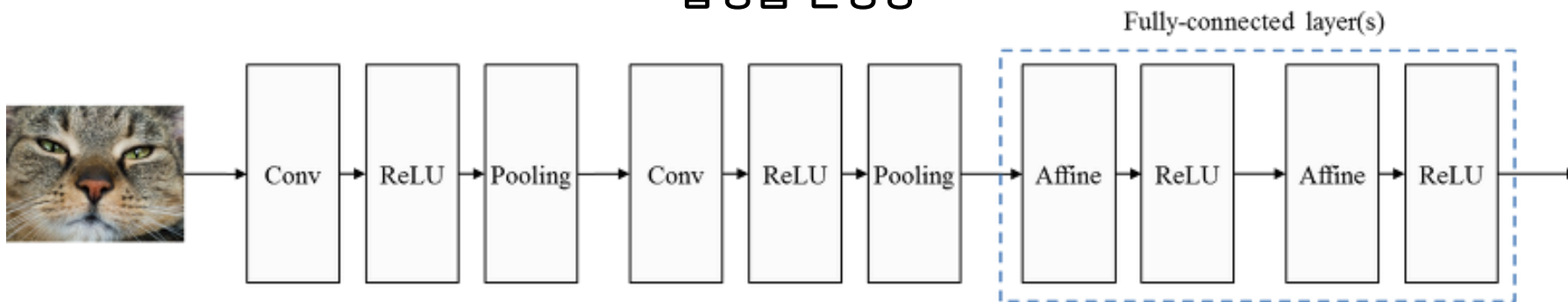
합성곱 층, 풀링 층, 완전 연결 층 등으로 구성

완전 연결 신경망

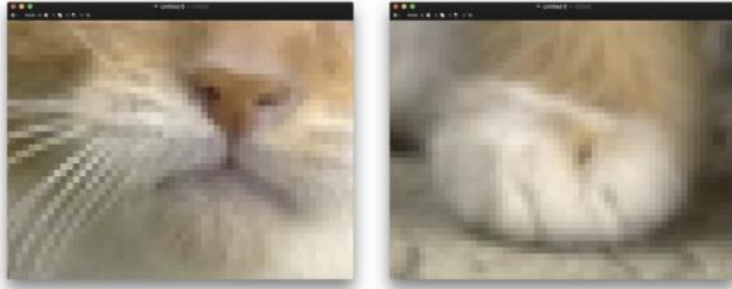


VS

합성곱 신경망



점과 선, 질감을 충분히 배우고, 조금 떨어져서 보자.



점과 선이 질감이 합쳐져 삼각형, 동그라미, 복실함이 보인다.

출처: 하용호님 자료 @kakao

삼각형, 원, 사각형, 복실함을 조합해서 보니



뽀족귀와 땡그란눈과 복실한 발을 배웠다.

출처: 하용호님 자료 @kakao

핵심 특징:

합성곱 층은 지역 패턴을 학습한다.
이미지일 경우 작은 2D window로 입력에서 패턴을 찾음

Ex) 이미지는 에지(edge), 질감(texture) 등의 지역 패턴으로 분해됨

<공간적 계층 구조를 학습>

1st - edge 같은 작은 지역 패턴을 학습

2nd - 첫 번째 층의 특성으로 구성된
더 큰 패턴을 학습

...

복잡하고 추상적인 시각적 개념을 효과적으로 학습



Components of CNN

- Convolutional Layer
 - Filter
 - Stride
 - padding
- Pooling layer

tensorflow.keras.layers에 속해있는 Conv2D 클래스로 구현

filter padding

+ stride, kernel_initializer, kernel_regularizer...

Conv layer → 1 model = keras.Sequential()

Pooling layer → 2 model.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu',
3 input_shape=(28, 28, 1)))

4 model.add(layers.MaxPooling2D((2, 2)))

5 model.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu'))

6 model.add(layers.MaxPooling2D((2, 2)))

7 model.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))

8 model.add(layers.MaxPooling2D((2, 2)))

9 model.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))

10 model.add(layers.MaxPooling2D((2, 2)))

11 model.add(layers.Flatten())

12 model.add(layers.Dense(64, activation='relu'))

13 model.add(layers.Dense(10, activation='softmax'))

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

1. Convolutional layer

0	1	7	5
5	5	6	6
5	3	3	0
1	1	1	2

 \otimes

1	0	1
1	2	0
3	0	1

 =

40	

0	1	7	5
5	5	6	6
5	3	3	0
1	1	1	2

 \otimes

1	0	1
1	2	0
3	0	1

 =

40	32

0	1	7	5
5	5	6	6
5	3	3	0
1	1	1	2

 \otimes

1	0	1
1	2	0
3	0	1

 =

40	32
26	

0	1	7	5
5	5	6	6
5	3	3	0
1	1	1	2

 \otimes

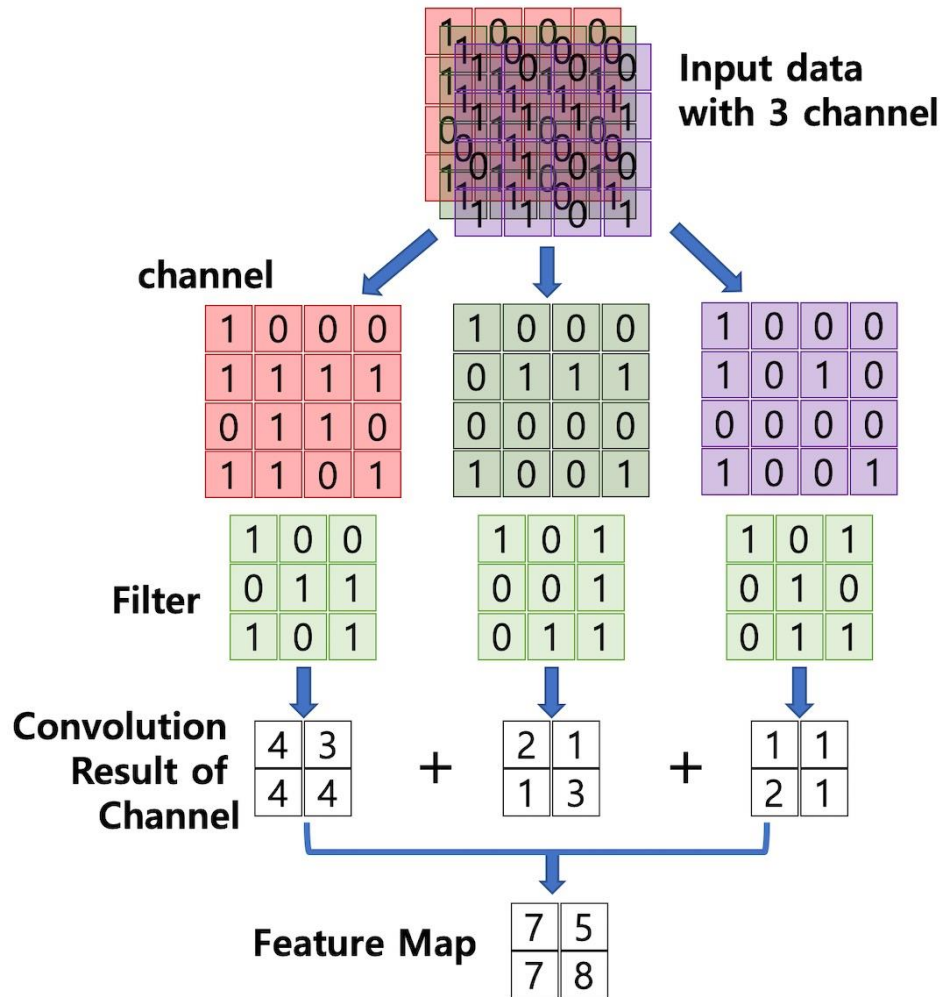
1	0	1
1	2	0
3	0	1

 =

40	32
26	25

- **Feature Map:** 입력데이터를 필터가 순회하며 합성곱을 계산하고, 그 계산 결과를 이용하여 위의 그림처럼 특징 맵을 만들어 이미지로부터 학습했던 내용을 표현하는 것
- **Activation Map:** Feature Map 행렬에 활성 함수를 적용한 결과 = Conv layer의 최종 출력 결과

1. Convolutional layer



Filter

- 이미지의 특징을 찾아내기 위한 공용 파라미터(W) - kernel이라고도 함
- 정사각행렬의 형태를 띠(그림에서는 (3, 3) 크기의 filter를 이용함)
- 지정된 간격으로 이동하면서 전체 입력데이터와 합성곱하여 Feature Map을 만듦

Stride

- "지정된 간격" → 그림에서는 한 칸씩 이동하고 있으니 stride = 1

1. Convolutional layer

0	1	7	5
5	5	6	6
5	3	3	0
1	1	1	2

 \circledast

1	0	0
1	2	1
1	2	3

 $=$

41	33
25	23

Padding 無

0	0	0	0	0	0
0	0	1	7	5	0
0	5	5	6	6	0
0	5	3	3	0	0
0	1	1	1	2	0
0	0	0	0	0	0

 \circledast

1	0	0
1	2	1
1	2	3

 $=$

26	42	55	35
34	41	33	28
18	25	23	14
3	9	8	8

Padding 有

Padding

Conv layer의 출력 데이터가 줄어드는 것을 방지하는 방법

→ 입력 데이터의 외각에 지정된 픽셀만큼 특정 값으로 채워 넣는 것(보통은 0으로 채움)

- padding='valid' : padding X, 즉 입력보다 출력의 크기가 작아짐
- padding='same' : padding 0, 입력과 출력의 크기는 같음

1. Convolutional layer

Conv layer의 출력 크기

$$\text{OutputHeight} = OH = \frac{(H + 2P - FH)}{S} + 1$$

(1)

$$\text{OutputWeight} = OW = \frac{(W + 2P - FW)}{S} + 1$$

(2)

※ 입력 데이터 높이: H, 입력 데이터 폭: W, filter 높이: FH, filter 폭: FW, stride 크기: S, padding 사이즈: P

>> 위의 식의 결과값은 자연수가 되어야 한다

>> Feature Map의 행과 열 크기는 뒤에 이를 pooling 층의 크기의 배수여야 한다

흔히 보는 **ValueError: Shapes are incompatible** 오류는
이 조건을 맞춰주지 않아 크기 호환이 안돼서 발생!!

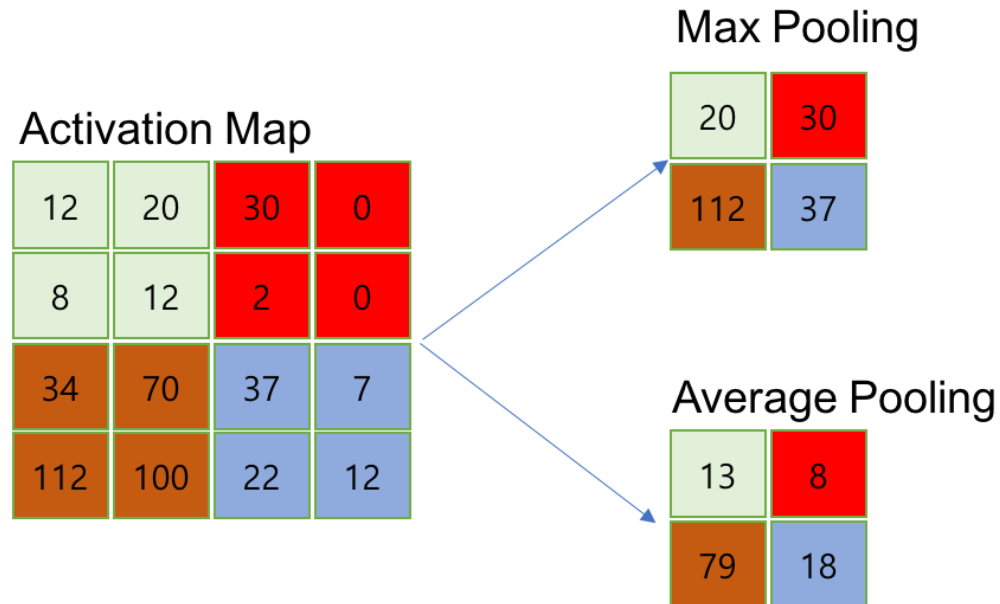
2. Pooling layer

Conv layer를 거쳐서 나온 activation map이 있을 때,
이를 이루는 Conv layer를 resizing하여 새로운 layer를 얻는 것
→ Activation Map의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용

최댓값 뽑기(MaxPooling)

평균값 뽑기(Average Pooling)

최솟값 뽑기(MinPooling)



2. Pooling layer



결국 pooling을 거치면 크기가 줄어든다. 그렇다면 pooling layer는 왜 쓰는걸까?

96x96의 input data(이미지 크기)가 있다고 가정해보자.

feature의 개수는 400개, 8x8의 filter를 사용한다고 하면, Conv layer을 지나온 후의 출력 크기(Activation Map)은 다음과 같음

$OH = OW = (96 - 8 + 1)$ 이므로,
크기는 $(96 - 8 + 1) * (96 - 8 + 1) = 7921$

feature가 400개 있으니까, 총 feature의 개수는 $7921 * 400 = 3,168,400$

이렇게 많은 feature 수를 classifier에 넣는 것은 overfitting을 유발시킨다!

그래서 pooling layer를 추가해주면 정해진 pixel 안에서 조건에 맞는 값을 뽑고, 크기를 조절할 수 있기 때문에 overfitting을 방지해줄 수 있음

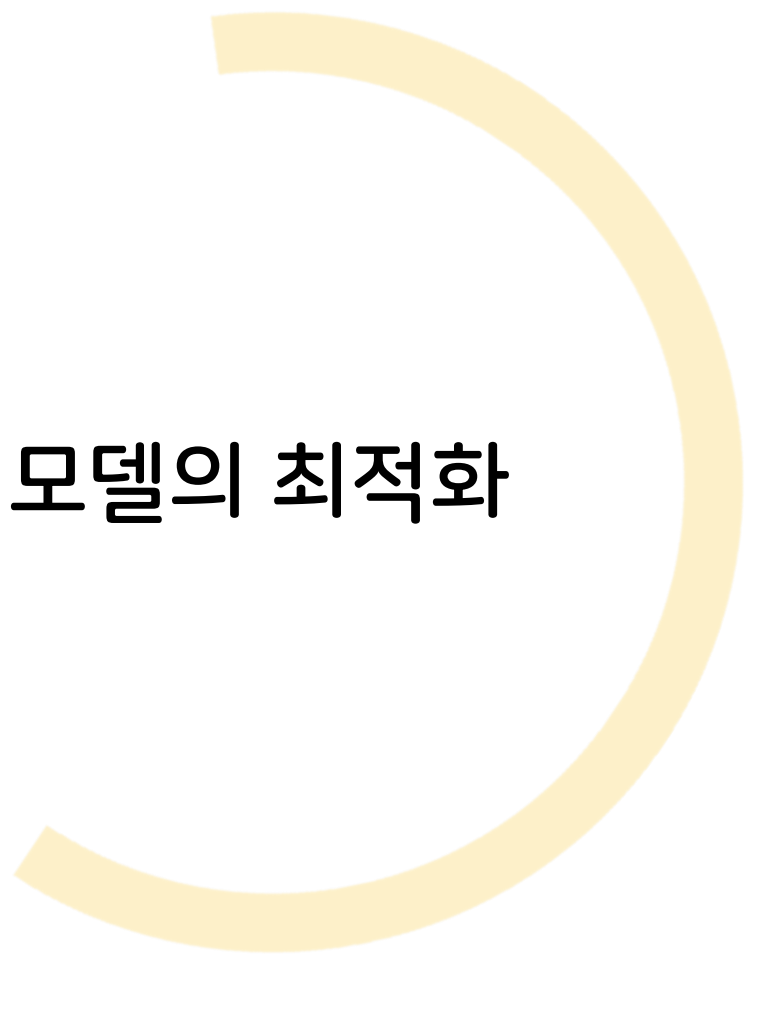
2. Pooling layer

Pooling layer의 출력 크기

$$(1) \quad OutputRowSize = \frac{InputRowSize}{PoolingSize}$$

$$(2) \quad OutputColumnSize = \frac{InputColumnSize}{PoolingSize}$$

Conv layer의 출력값 -> Pooling layer의 입력값
(배수 맞춰주기)



모델의 최적화

BatchNormalization
Dropout
Ensemble
등등

모델의 최적화

하이퍼 파라미터 조정

최적의 딥러닝 모델 구현을 위해 학습률이나 배치크기, 훈련 반복 횟수, 가중치 초기화 방법 등 인간의 선험적 지식을 기반으로 딥러닝 모델에 설정하는 변수

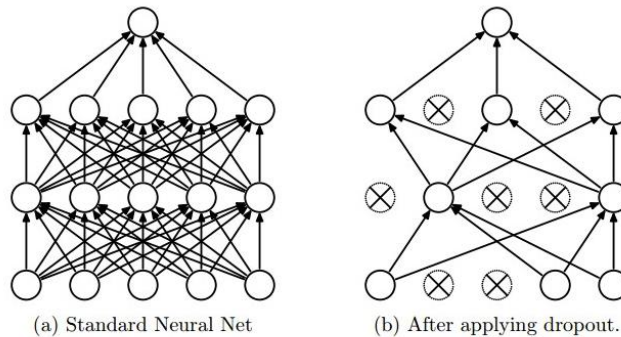
BatchNormalization?

머신 러닝 모델에 주입되는 샘플들을
균일하게 만드는 광범위한 방법
(정규화)

새로운 데이터에 잘 일반화 되도록!

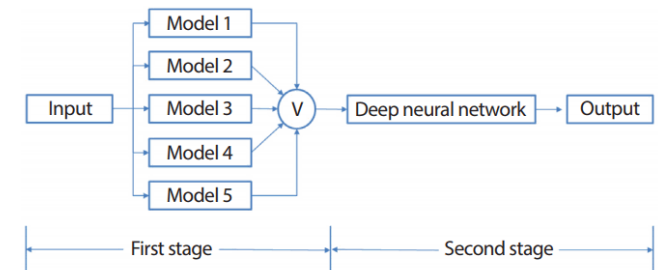
Dropout?

훈련할 때 임의의 뉴런을 골라 삭제하여
신호를 전달하지 못하도록 하는 방법



Ensemble?

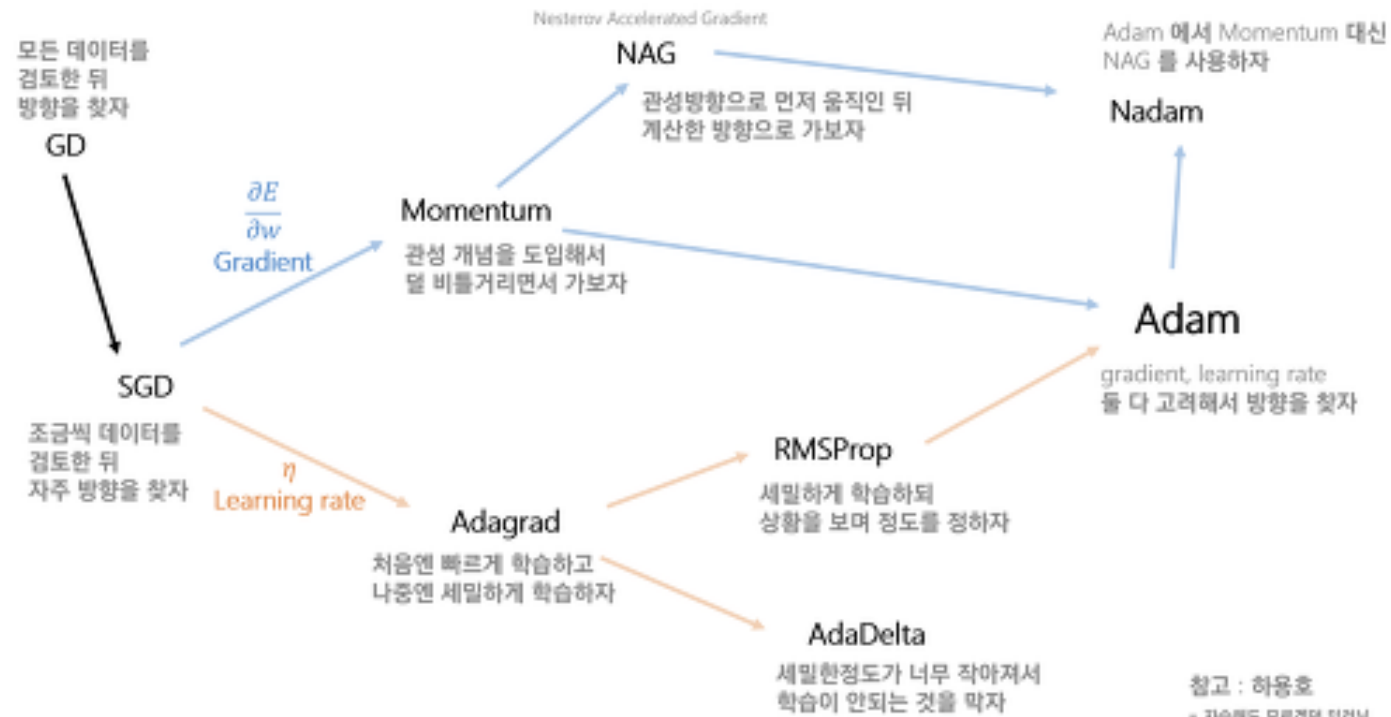
여러 개 다른 모델의 예측을 합쳐서
더 좋은 예측을 만드는 방법



모델의 최적화

Optimizers?

<https://keras.io/api/optimizers/>



Convolutional Neural Network



실습